2011 American Control Conference
on O'Farrell Street, San Francisco, CA, USA
June 29 - July 01, 2011

Multi-Runway Aircraft Sequencing at Congested Airports

Luc Bojanowski, Dimitri Harikiopoulo and Natasha Neogi

***Abstract*— In this paper, we consider the airport landing problem of scheduling aircraft on multiple runways in a dynamic fashion. We attempt to modify the aircraft landing sequence from the traditionally used "First-Come-First-Served" ($FCFS$) order to be able to land more aircraft in a given period of time. Given a set of planes, the goal is to find a sequence such that no plane can land before it is actually available for landing, the minimum safety separation between two consecutive planes is always satisfied, and that the total landing time (makespan) is minimized. Based on the FAA partition of aircraft into weight class, our algorithm, based on previous work, provides a polynomial time feasibility condition for scheduling a set of planes in a given time interval. This algorithm for the dynamic scheduling of aircraft in a multi-runway scenario is factorial in runway number and exponential in aircraft class, both of which are fixed *a priori*, however it is polynomial in the number of aircraft, which is the dominating factor in the problem. It ensures that the Aircraft Sequencing Problem (ASP) presented above is not NP-Complete and allows us to develop a practical real time ATC execution policy.**

I. Introduction and Motivation

The incessant increase in air traffic over the past few decades in the face of limited resources and infrastructure has lead to the proliferation of flight delays at major airports, which cost the U.S. airline industry an estimated 5.5 billion dollars annually. It has been estimated by the FAA that, by the year 2015, if the air traffic control (ATC) infrastructure and operating procedure does not change, there could be an aviation accident once every seven to ten days. Hence, the need to develop improved ATC strategies at over-saturated airports must necessarily be subject to safety critical constraints that maintain current standards in aircraft separation and controller workload. Therefore, any ATC strategy should respect the minimum separation requirements and provide results quickly enough to allow the controller to make the right decisions at the right time.

A. First Come, First Serve Ordering

In order to prevent a trailing aircraft from losing aerodynamic stability because of turbulence (wake vortices) generated by a leading aircraft, the Federal Aviation Administration (FAA) has fixed minimum separation requirements between aircraft. Thus, to guarantee safety, the FAA has generally partitioned aircraft into three weight classes: Small, Large, Heavy; and fixed in-trail separation requirements. The time separation requirements are then a function of the plane speed and the length of the final approach path. An example for a 6 miles long final approach path is given in Table I, and was derived in Odoni [9],[5]. We use these numerical values for time separation in our simulations.

	Small	Large	Heavy	Speed (kts)
Small	82	69	60	110
Large	131	69	60	130
Heavy	196	157	96	150

TABLE I
TIME SEPARATION REQUIREMENTS (IN S) AND CLASSES SPEED (IN KNOTS)

If we consider one unique class, then the separation times between all successive aircraft are equal, since all planes of the same weight class should enter the Terminal Maneuvering Area (TMA) at the same speed. In this case, the "First Come, First Served" (FCFS) landing order (that is, ordering with respect to Earliest Landing Time (ELT)), ensures an optimal landing time for the whole set of planes. Now if we consider several classes, the FCFS strategy, despite being fair and easy to implement, is no longer optimal due to the assymetry in the separation requirements, underlined above. That said, if we consider a particular landing sequence, applying the FCFS strategy within each class minimizes the makespan, corresponding to this particular sequence.

B. Improving Upon FCFS

There are three ways to define the optimality of a scheduling strategy:

1) Minimization of the sum of Landing Processing Times (LPT)
2) Minimization of the overall Makespan
3) Minimization of the Last Aircraft of the sequence 's LST

For each aircraft, the LPT is the difference between its Landing Completion Time (LCT) and its Landing Starting Time (LST), i.e. the required separation time between the considered aircraft and the following aircraft.

It is of note that minimizing the sum of LPTs is not equivalent to minimizing the makespan nor equivalent to minimizing the last aircraft's LST. However, if there is no idle time present, minimizing the makespan and the sum of completion times become equivalent.

In this paper we develop an algorithm which solves the dynamic scheduling problem for a multi runway airport for

Luc Bojanowski is with Proctor and Gamble SAS, 163-5 quai Aulagnier, 92600 Asnieres Sur Seine, France, lbojanow@uiuc.edu
Dimitri Harikiopoulo is with the Thales Group, 45 Rue de Villiers, 92526 Neuilly-sur-Seine Cedex, France, dimitri.harikiopoulo@thalesgroup.com
Natasha Neogi is with the National Institute for Aerospace, 100 Exploration Way, Hampton, VA, 23666, USA, NatashaNeogi@nianet.org

an optimized makespan over a constrained interval. The algorithm is a novel contribution in that while it is exponential in aircraft class, it is merely factorial in runway number, and polynomial in aircraft number. We place this result in context of the current state of the art, in the next section.

II. Previous work

The Aircraft Sequencing Problem (ASP) has received a great deal of attention and its static and dynamic versions have been studied using various approaches and techniques described in the literature [6], [7], [1], [9], [11]. Dear's paper [6], and the one written with Sheriff [7], introduced the constrained position shifting (CPS) technique. This technique schedules planes such that their final position does not differ from the $FCFS$ order by more than a pre specified number, called Maximum Position Shifting (MPS). They applied this technique to both the static and the dynamic version of the Aircraft Scheduling Problem (ASP) through an heuristic algorithm and presented computational results involving up to 500 planes.

Psaraftis [1] considers the static problem where all the planes are available at time $t = 0$ and the minimum separation are sequence dependent only throughout the plane classes. He solves it exactly with a dynamic programming backward recursion .The algorithm is then improved by adding a feasibility condition to his recursion such that the CPS constraint is satisfied. Taking advantage of the class repartition, this algorithm finds an optimal sequence in polynomial time. This article is based on a previous one [8], where Psaraftis also considers the two-runway problem. Venkatakrishan, Barnett and Odoni [9] built a statistical model for the time separation between landing based on real data observed at Boston Logan Airport. They use this model and a heuristic modification of Psaraftis algorithm for time windows constraint (earliest and latest possible landing times) to improve landing aircraft sequencing. Three models are presented, one for the static case, two for the dynamic case and their results, tested on real data, show that sequencing improvements can reduce delays by 30 % in some instances. Based on the same Psaraftis' algorithm [1], our work focuses more on the theoretical aspect and aims at proving the aircraft landing problem with time window constraint is not NP-hard.

A more generic instance of the static case of the aircraft sequencing problem is explored by Beasley et al [10]. Instead of dealing with planes partitioned into classes, they consider all potentially different aircraft that are required not only to land in a prescribed time window but also to satisfy complete separation requirements. Complete separation means the separation does not only apply to successive aircraft but to all pairs of aircraft. They give a mixed integer zero-one formulation of the problem for both the single and multiple runway problem and provide some results involving up to 50 planes.

Another very interesting approach can be found in the work of Bianco et al [11], where they solve a single-machine scheduling problem with "release times" equivalent to the one runway aircraft landing problem. They formulate it using a mixed integer zero-one formulation and solve it with both a branching algorithm and a heuristic procedure. In some later work, Bianco, Dell'Olmo and Giordani [4], [12], keep this machine scheduling approach but try to minimize the sum of completion time instead of the maximum completion time. The problem is solved using dynamic programming, lower bounds and heuristic algorithms. Computational results are presented for sets of 30 and 44 planes. In these papers the algorithms are derived considering that all planes are potentially different and the class repartition is only used for the numerical simulation. This differs from our approach, where we exploit the class partition to simplify the theoretical algorithm and then reduce its complexity.

All these papers model the landing problem as single resource models since an arriving aircraft needs one runway to land. In Bianco, Dell'Olmo and Giordani [13], the authors present a job shop scheduling approach dividing the approach path into discrete segments and representing each of them by a machine. This TMA representation as a multiple resource problem allows them to represent the evolution in time of aircraft positions in the TMA. The problem is solved by local search techniques, and results based on real data sets from Rome Fiumicino TMA are provided.

Much work not directly related to our approach has also been done. Queuing theory models [14] have been developed as well as decision aid software, like $CTAS$ (Center $TRACON$ Automation System), currently in use in many airports. For more references, an extended literature survey can be found in Beasley et al [10] and a broad state of the art compilation of research in Air Traffic Management is found in Wu [15].

Harikiopoulo proposed an algorithm solving the ASP in polynomial time in the case of a single runway airport and suggested its generalization to the multiple runways case [2], upon which the following work is based. We refer the reader to [3] for the mathematical details of this algorithm.

A. Recursive Algorithm Extension for Multiple Runways

We consider the single runway state space to be described by $(m, \boldsymbol{k})$, where:

- m Class of the plane currently landing on the runway
- $\boldsymbol{k}$ Class repartition vector, i.e. the p-component vector (p being the number of classes of aircraft) of the number of aircraft of each class. For instance, if $\boldsymbol{k} = (2, 1, 3)$, then it means there are two planes of class 1, on plane of class 2 and three planes of class 3 still remaining to land.

To incorporate M runways in the description of the problem state space, we replace $\boldsymbol{m}$ in the state vector by $(m_1, \ldots, m_M)$, where m_i is the class currently landing on runway i. We denote $\boldsymbol{V}((m_1, \ldots, m_M), (k_1, ..., k_p))$ as the minimum total landing time to land an aircraft of class m_i on runway i for $i \in \{1, \ldots, M\}$ with k_j aircraft of class j, for $j \in \{1, \ldots, p\}$, still remaining to be landed. Vectors $\boldsymbol{V}((m_1, \ldots, m_M), (k_1, ..., k_p))$ and $\boldsymbol{t}((m_1, \ldots, m_M), (k_1, ..., k_p))$ now have M components

since they include the minimum landing times for all runways. For example, for two runways and three planes, of which two are $Small$ and one is $Large$, where initially the first runway has just landed a $Small$ plane, and the second has also landed a $Small$ plane, the formulation for the initial configuration is as follows:
$M = 2$ (number of runways)
$\boldsymbol{m} = (m_1, m_2)$, where $m_1, m_2 \in \{Small, Large, Heavy\}$
$\boldsymbol{m_0} = (Small, Small)$ (initial state)
$\boldsymbol{k} = (k_{Small}, k_{Large}, k_{Heavy})$
$\boldsymbol{k^0} = (2, 1, 0)$
$\boldsymbol{V}((m_1, m_2), (k_{Small}, k_{Large}, k_{Heavy}))$
$= \boldsymbol{V}((Small, Small), (2, 1, 0))$
$= \boldsymbol{V}(\boldsymbol{m_0}, \boldsymbol{k^0})$
Note that the multi runway formulation increases the dimension of the problem by converting the scalar m from the single runway formulation into a vector.

With all the aforementioned modifications, and defining the infinite norm of a vector $\boldsymbol{x} = (x_1, \ldots, x_n)$:

$$\forall \boldsymbol{x} \in \mathcal{R}^n, ||\boldsymbol{x}||_\infty = max_{1 \leq i \leq n} |x_i| \tag{1}$$

the recursive relation of Harikiopoulo's single runway algorithm becomes:
$\boldsymbol{V}((m_1, \ldots, m_M), (k_1, ..., k_p))$

$$= \begin{cases} \boldsymbol{t}((m_1, \ldots, m_M), (k_1, ..., k_p)) & \text{if } feas = 1 \\ (+\infty, \ldots, +\infty) & \text{if } feas = 0 \end{cases} \tag{2}$$

where

$$\boldsymbol{t}((m_1, \ldots, m_M), (k_1, ..., k_p)) = \begin{cases} (0, \ldots, 0) \text{ if } k_1 = ... = k_p = 0 \\ \min_{\substack{x \in X \\ y \in \{1, ..., M\}}} ||\boldsymbol{f}((m_1, \ldots, m_M), x) \\ \quad + \boldsymbol{V}((m'_1, \ldots, m'_M), (k'_1, ..., k'_p))||_\infty \text{ otherwise} \end{cases} \tag{3}$$

$$\boldsymbol{f}((m_1, \ldots, m_M), x) = (0, \ldots, 0, c(m_y, x), 0, \ldots, 0) \tag{4}$$

$$m'_i = \begin{cases} x, \text{ if } y = i \\ m_i \text{ otherwise} \end{cases} \tag{5}$$

$$X = \{x | k_x > 0\} \tag{6}$$

and

$$k'_i = \begin{cases} k_i - 1, \text{ if } i = x \\ k_i \text{ otherwise} \end{cases} \tag{7}$$

$$feas = 1 \Leftrightarrow D - \boldsymbol{t}((m_1, \ldots, m_M), (k_1, ..., k_p)) \geq ELT(m_y, k''_{m_y}) \tag{8}$$

where equation (8) defines the criterion for constructing a feasible schedule.

However, it is now necessary to keep track of the number of planes of each class remaining to land and their corresponding ELTs, on a per runway basis. We introduce a class repartition vector for each runway, each of them being updated when an aircraft lands on the corresponding runway. The calculation of the LSTs of the planes of the sequence then becomes:

- ***S1* lands on *R1*:** As no plane of class $Small$ landed before, we set $\boldsymbol{k_{R1}}$ to $\boldsymbol{k_{R1}} = \boldsymbol{k^0}$. $S1$ is then identified by $\boldsymbol{k_{R1}}(1)$. We have: $LST(S1) = ELT(S1) = ELT(Small, \boldsymbol{k}(1)) = ELT(Small, 2)$.
- ***S2* lands on *R2*:** As one plane of class $Small$ already landed before, we remove one plane of class $Small$ from the class repartition vector $\boldsymbol{k}$, so we set $\boldsymbol{k_{R2}}$ to $\boldsymbol{k_{R2}} = (1, 1, 0)$. $S2$ is then identified by $\boldsymbol{k_{R2}}(1)$. We have $LST(S2) = ELT(S2) = ELT(Small, \boldsymbol{k}(1)) = ELT(Small, 1)$.
- ***L1* lands on *R1*:** As there was a plane of class $Small$ on $R1$ before, then we have:

$$\begin{aligned} LST(L1) &= max(ELT(L1), LST(Small, \boldsymbol{k_{R1}}(1)) \\ &\quad + c(Small, Large)) \\ &= max(ELT(Large, \boldsymbol{k_{R1}}(2)), \\ &\quad ELT(Small, \boldsymbol{k_{R1}}(1)) \\ &\quad + c(Small, Large)) \\ &= max(ELT(Large, 1), ELT(Small, 2) \\ &\quad + c(Small, Large)) \\ &= max(ELT(Large, 1), ELT(S1) \\ &\quad + c(Small, Large)) \end{aligned}$$

which is correct as it associates $S1$'s ELT to $R1$.

As a result, in equation (8), k''_{m_y} is not necessarily equal to k_{m_y}. Instead, with the aforementioned notations, we have:

$$k''_{m_y} = \boldsymbol{k_y}(m_y) \tag{9}$$

B. Induction vs. Recursion

Our modifications to the outlined recursive approach is motivated by the following key factors:

LST of initial aircraft: The standard single runway algorithm considers that the initial plane (class m_0) is always available for landing and then sets $LST(m, \boldsymbol{k^0_m}) = -\infty \quad \forall m \in \{1, \ldots, p\}$. As a result, the first column of **V** is filled with zeros only. We decide to take into account the LST of the 0^{th} plane on each runway for two main reasons:

1) This allows us to use the algorithm for the dynamic case in which the optimal sequence will be built in dynamics increments, where new aircraft with ever greater ELTs are continuously arriving;
2) In the multiple runways case, all the 0^{th} planes have not necessarily begun landing at the exact same time. Therefore, not considering the real values of the 0^{th} planes' LSTs would introduce an offset in runway utilization between runways, that could lead the algorithm to return a sub-optimal sequence.

Initially free runways: In the single runway problem, we consider the case where $m_0 \in \{1, \ldots, p\}$ or where $m_0 = 0$ (meaning that there is no plane in the previous sequence)

simultaneously. In the multiple runways case, we always consider an $\boldsymbol{m_0}$ vector where

$$m_0(i) \in \{1, \ldots, p\} \forall i \in \{1, \ldots, M\}$$

for the following reasons:

1) This is the simplest way to distinguish between two configurations where the state and class repartition vectors are the same but corresponds to different 0^{th} planes.
2) It is a way to easily eliminate the unreachable states in the $\mathbf{V}$ array, thereby avoiding the calculation of unnecessary values and the loss of calculation capacity. If there are one (or more) initially free runways, we select the requisite number of planes from those that are available to land in order to fill the $\boldsymbol{m_0}$ vector. This is supported by the introduction of the ELTs of the 0^{th} plane on each runway; as now we can consider the subproblem of selecting $\boldsymbol{m_0}$ among the possible combinations to yield the optimal schedule.

LSTs vs LCTs: The recursive algorithm fills the $\mathbf{V}$ array with the completion times of the previous plane in the constructed sequence; that is the earliest time at which the runway is free for landing an aircraft. Thus, we fill the elements of the $\mathbf{V}$ array with the vector of LSTs corresponding to the last aircraft in the sequence constructed to that point, on each runway. Thus, each element of $\mathbf{V}$ has M components, and for each valid landing sequence, there is a corresponding element of $\mathbf{V}$. [1] We are interested in minimizing the LST of the last plane of the sequence (the maximum of the LSTs of the last aircraft landing on each runway), and not the makespan. As a result, (3) now becomes:

$\boldsymbol{t}((m_1, \ldots, m_M), (k_1, ..., k_p)) =$

$$\begin{cases} (0, \ldots, 0) \text{ if } k_1 = ... = k_p = 0 \\ \min_{x \in X} \quad \max(ELT(x, k_x), \\ \quad y \in \{1, ..., M\} \\ \quad ||\boldsymbol{f}((m_1, \ldots, m_M), x) \\ \quad\quad +\boldsymbol{V}((m'_1, \ldots, m'_M), (k'_1, ..., k'_p))||_\infty) \text{ otherwise} \end{cases} \tag{10}$$

where $ELT(x, k_x)$ is the ELT of the plane we currently want to land, i.e. the k_x^{th} plane of class x and equations (2, 4, 5, 6, 7, 8) are still valid.

Dynamic Sequencing via Induction: We build the sequence inductively for several reasons:

1) Presence of unreachable configurations in the $\mathbf{V}$ array: Some configurations of $\mathbf{V}$ are not reachable because the state vector and the class repartition vector allocate more planes of a given class than are actually present in the initial state vector $\boldsymbol{m_0}$ and the initial class repartition vector $\boldsymbol{k^0}$. These configurations are easy to identify. But there exist other unreachable configurations, corresponding to state and class repartition vectors satisfying the previous condition but for which no predecessor is a reachable configuration. To identify these configurations, we make a preliminary check of the elements of $\mathbf{V}$ to identify all configurations that do not allocate more planes than are initially present. We then make a second check to prune out all valid configurations that have no reachable predecessors. For instance, if

$$\begin{aligned} \mathbf{m_0} &= (Small, Small) \\ \mathbf{k^0} &= (2, 1, 0) \end{aligned}$$

then the configuration $\boldsymbol{V}((Small, Small), (2, 0, 0))$ seems at first to be reachable because $\boldsymbol{m} = (Small, Small)$ and $\boldsymbol{k} = (2, 0, 0)$, i.e. contains four planes of class $Small$, not more than initially (four planes of class $Small$ and one plane of class $Large$). However, the only possible predecessor class repartition vector is $(3, 0, 0)$. That is because the state vector $\boldsymbol{m}$ consists only of Small aircraft remaining to land; thus on the predecessor configuration, a Small plane must have landed, requiring $\boldsymbol{V}((Small, Small), (3, 0, 0))$ to preceede $\boldsymbol{V}((Small, Small), (2, 0, 0))$.
2) Lack of sufficient information to uniquely identify (landed aircraft, landing runway) pairs:
When calculating a particular $\boldsymbol{V}((m_1, \ldots, m_M), (k_1, ..., k_p))$ using 2, 10, 4, 5, 6, 7 and 8, we need to consider the correct ELTs of each $(aircraft, runway)$ pair. This is not easy because while building the V array recursively, as we know how many aircraft of the same class have landed previously but we cannot tell how many aircraft of the same class have landed previously on a given runway. This can lead to incorrect values in $\mathbf{V}$ or make the algorithm more complicated.

C. Dynamic Optimality in the Multiple Runways Case

In reality, aircraft arrive dynamically to the TRACON. Each aircraft is usually assigned to a particular runway a significant time before it actually lands. This means that once it has been assigned to a particular runway, it may not land on another, but its ELT can still be subject to some slight modification (either an advance or a delay). These fluctuations can lead to a sequence that is no longer optimal. To adhere as much as possible to the optimal sequence, we apply several levels of optimization to the set of aircraft to land, and divided it into three proper subsets:

- Aircraft not yet assigned to a particular runway
- Aircraft assigned to a particular runway but whose position on the runway is still subject to modifications
- Aircraft that will no longer be resequenced.

Each time the algorithm is run, we then:

- Update the ELTs of the aircraft not belonging to the third subset
- Apply the single runway algorithm to the aircraft of the second subset
- Apply the multiple runways algorithm to the aircraft of the first subset

[1] Please refer to [2], section 'V.Example' for an explicit example of how to build the $\mathbf{V}$ array.

in this order.

This allows us to obtain a sub-optimal sequence in a realistic way, taking into account the fact that the aircraft must have prior knowledge of their landing runway, in order to enter the TMA at the corresponding entry point.

III. Dynamic Runway Scheduling Problem

A. General Structure of the Inductive Algorithm

Using our inductive reformulation and introducing the LSTs of the 0^{th} planes, we obtain a pseudo-optimal landing sequence, in term of the largest of the LSTs of the last aircraft landing on each runway, of a set of planes, in four main steps.

Step 1: Benchmark the FCFS sequence: Calculate the makespan of the FCFS sequence and then set B, the time taken to land a feasible sequence, to:

$$B = \text{LST of the last plane of the FCFS sequence}$$

Step 2: Define a 0^{th} plane:

- If an aircraft of class i is initially on the runway, set $\boldsymbol{m_0} = (i)$ and then go to Step 3.
- If the runway is initially free, separate the search for the optimal sequence into as many subproblems as classes of aircraft. For each subproblem, the initial aircraft is of a different class, and for each given subproblem, go to Step 3.

Step 3: Finding a feasible sequence: Calculate a feasible sequence within a time B for the given set of planes with the class of the plane on the runway at $t = 0s$ being given as $\boldsymbol{m_0}$, by identifying all reachable configurations in $\boldsymbol{V}$ defined in II and picking the optimal. We can iterate to find decreasing values of B via the procedure explained in Step 4.
In case several configurations provide the same LST, we choose the configuration according to the following priority policy:

1) We give priority to the configuration minimizing the difference between the ELT of the plane we want to land on the runway and the LCT of the plane that landed before on this runway.
2) We then give priority to the smaller class number, to avoid incurring the worst future penalty.

Step 4: Obtaining the static optimal sequence: Two methods are used successively in order to find the optimal value of B possessing a feasible sequence. These methods were described in [2].

DICHOTOMY: An upper bound UB is given by LST of the last plane of the FCFS sequence.A lower bound LB is given by either the ELT of the last available aircraft or by the LST of the last aircraft of the optimal sequence for all ELTs equal to zero. We halt the dichotomous search, specified in [2], when $UB - LB \leq \epsilon$, where ϵ is a pre-set threshold.

DECREMENTAL METHOD: Once $UB - LB \leq \epsilon$, we start with the last UB of the dichotomy process which returned a feasible sequence, and we decrement the upper bound by a single unit until the algorithm can no longer provide a feasible sequence. The method is detailed in [2].

Note: There can exist several different feasible sequences for the optimal value of B. However, the algorithm returns one unique solution based on the priority rules invoked in Step 3.

IV. Multiple Runways Scheduling Problem

The single-runway algorithm presented in the previous section is a restriction of our algorithm. This section shows how considering several runways affects the algorithm and its performances.

A. First Come, First Serve

For the multi-runway case, getting the FCFS sequence not only means ordering all the aircraft according to their ELTs, but also choosing for each aircraft the runway allowing it to land as soon as possible, i.e. the runway minimizing its LST. If each LST is minimized with respect to the FCFS ordering, the LST of the last plane of the FCFS sequence is also minimized.

As in the single runway case, if we impose a landing sequence, optimality will be reached if the FCFS ordering is respected within each class.

B. Improving upon FCFS

In the multiple runways case, we can distinguish the LST of the last aircraft of each runway and the global LST of the set of planes, i.e. the maximum of each runway's LST. As our goal is to land the maximum number of planes in a minimum time, we are only interested in the latter and will therefore call it simply the LST of the last plane hereon.

C. Algorithmic Approach

Considering several runways affects the general structure of our algorithm as follows:

1) Step 1: FCFS sequence: As presented above, calculate the optimal FCFS sequence (per one or M runways).

2) Step 2: Defining 0^{th} planes: As we have several runways, we can imagine three situations:

- All the runways are occupied with an aircraft at $t = 0$
- All the runways are free at $t = 0$
- Some runways are free and some others are occupied at $t = 0$. As in the single runway case, we need to consider all the possibilities for the first plane on each runway and for each case go to Step 3.

3) Step 3: Finding a feasible sequence: Ambiguity arises when we want to extend the feasibility algorithm to the multiple runway problem: in the single runway problem, an aircraft and its associated ELT is exactly determined by the sole knowledge of its class m and the number k_m of aircraft remaining to be landed within class m. This is true because $k_m^0 - k_m$ provides the arriving position of the aircraft within its class. In the multiple runway problem, the aircraft class, denoted m, and the number k_m of aircraft remaining to be landed within this class, may not be sufficient to exactly associate an aircraft and its corresponding ELT. In order to be able to exactly retrieve the position of each aircraft, we need to differentiate all the possible landing

orders and include them in the state representation. This state extension increases the Running Time Complexity (RTC) and the Storage Complexity (SC). Harikiopoulo demonstrated in [3] that:

$$RTC = M * p * F(p, M) * \prod_{i=1}^{p}(1 + k_i^0) \tag{11}$$

and

$$SC = F(p, M) * \prod_{i=1}^{p}(1 + k_i^0) \tag{12}$$

where $F(p, M)$ is the number of possible values taken by $(m_1, \ldots, m_m)$. If we do not consider the state extension problem highlighted above:

$$F(p, M) = p^M \tag{13}$$

Now if we consider the state extension problem, we calculated that then:

$$F(p, M) = M! \sum_{i_1=1}^{M+1} \sum_{i_2=1}^{i_1} \sum_{i_3=1}^{i_2} \cdots \sum_{i_{p-1}=1}^{i_{p-2}} 1 \tag{14}$$

However, as said in I.I-A, we always follow a FCFS strategy within each class. Thus, if the state vector $\boldsymbol{m}$ contains several ELTs corresponding to the same class, we know that among the corresponding aircraft, the aircraft which landed first is the aircraft with the smallest ELT. As a result, state explosion is mitigated.

Finally, we add a third priority rule for the choice of a feasible sequence, in case B leads to several feasible sequences with the same largest LST: Minimize the ordinality of the runway identifier with the largest LST.

4) Step 4: Obtaining the pseudo-optimal sequence:

DICHOTOMY: We can easily evaluate a lower bound and an upper bound for the optimal LST of the last plane of the sequence, using the FCFS benchmark, and then use the procedure outlined in Section III A, Step 4.

DECREMENTAL METHOD: Using the last UB of the dichotomy process that returned a feasible sequence, we decrement the upper bound by a single unit until the algorithm cannot provide a feasible sequence.

Formally, from equations (11),(12),(14) the algorithm is factorial in the number of runways, i.e. $O(M!)$, exponential in the number of classes, i.e. $O(N^P)$, but polynomial in the number of aircraft in each class, i.e. $O(\prod_{i=1}^{p}(1+N))$. It also provides an optimal makespan for the constrained dynamic interval over which the algorithm is iteratively run.

D. Example Simulation Scenarios and Algorithm Performance

In this subsection, we consider the case of two classes of aircraft, $Small$ and $Large$, landing on three runways.

1) Simulation Parameters: In the following simulations, we consider a three runway airport and generate the set of planes and their ELTs according to a Poisson process with a 165 planes/hour rate of arrival, which is a realistic representation of a congested airport.

2) Time saved in comparison to a FCFS strategy: The following results are the average results obtained with a number of aircraft $N = 30$ over 25 simulations for each case. When the number of classes p is strictly superior to 1, any $\mathbf{P} = \begin{pmatrix} p_1 & p_2 & \ldots & p_p \end{pmatrix}$, where p_j is the proportion of planes of class j, is considered as a different case. The values presented in the tables below corresponds to the LST of the last plane of the FCFS sequence divided by the LST of the last plane of the optimal sequence. In all cases, we have supposed
$\mathbf{m_0} = \begin{pmatrix} Small & Small & Small \end{pmatrix}$ and
$\mathbf{0^{th}\_LST} = \begin{pmatrix} 0 & 0 & 0 \end{pmatrix}$

Single class: As expected, the FCFS is the optimal sequence.

Two classes: We consider the planes of class Large and Heavy use the following separation time matrix $\mathbf{C}$:
$\mathbf{C} = \begin{pmatrix} 69 & 60 \\ 157 & 96 \end{pmatrix}$
and we modify the proportion of planes of class Large by steps of 0.2. The proportion of planes of class Heavy is $p_{Heavy} = 1 - p_{Large}$.

p_L	Value
0.0	1.00
0.2	1.06
0.4	1.12
0.6	1.11
0.8	1.10
1.0	1.00

Three classes: We use the separation times given in I-A and we modify the proportion of planes by steps of 0.2. The rows of the matrix corresponds to the proportions of planes of class Small and the columns to the proportions of class Large. The proportion of planes of class Heavy is $p_{Heavy} = 1 - p_{Small} - p_{Large}$.

$p_S \backslash p_L$	0.0	0.2	0.4	0.6	0.8	1.0
0.0	1.00	1.06	1.12	1.07	1.05	1.00
0.2	1.12	1.15	1.13	1.15	1.07	
0.4	1.15	1.17	1.20	1.10		
0.6	1.12	1.21	1.07			
0.8	1.10	1.05				
1.0	1.00					

3) Computation Time and Resulting Trends: Harikiopoulo proved in [2] that the multiple runway aircraft scheduling problem is not NP-complete. However the run time can be computationally prohibitive as the number of classes or number of runways increases.

The computation times shown in the tables below were obtained with the algorithm implemented in Matlab on a Pentium IV. The results corresponds to average values over 25 simulations in each case, and in all cases:
$\mathbf{m_0} = \begin{pmatrix} Small & Large & Heavy \end{pmatrix}$ and

$\mathbf{0^{th}\_LST} = \begin{pmatrix} 0 & 0 & 0 \end{pmatrix}$.

Influence of the number of aircraft: Using our scenario parameters, with p=3 (three classes) and M=3 (three runways), the following computation times were obtained for the inductive algorithm:

Number of aircraft	Calculation time in s
3	0.25
4	0.49
5	0.98
10	11.0
20	129
30	557
40	983

Influence of the number of classes: Using our scenario parameters, with N=30 (thirty aircraft) and M=3 (three runways), we observe the following influence of the number of classes of aircraft on the computation time. We consider the following matrix **C'**:

$$\mathbf{C}' = \begin{pmatrix} 144 & 138 & 101 & 85 & 61 \\ 198 & 143 & 122 & 88 & 77 \\ 200 & 160 & 132 & 111 & 89 \\ 202 & 196 & 178 & 131 & 94 \\ 202 & 198 & 181 & 175 & 104 \end{pmatrix}$$

and suppose that for each number of classes p to consider (with $p \in \{1, \ldots, 5\}$), the seperation time matrix to consider is the $p*p$ upper left part of **C'**. For instance, if $p = 3$, then:

$$\mathbf{C} = \begin{pmatrix} 144 & 138 & 101 \\ 198 & 143 & 122 \\ 200 & 160 & 132 \end{pmatrix}$$

The following results were obtained:

Number of classes	Calculation time in s
1	0.26
2	21.7
3	418
4	4003
5	$\geq$ 75 000

V. Conclusion

An inductive algorithm for scheduling N aircraft, all of which are drawn from P classes of aircraft, onto M runways, in both a static and dynamic fashion. Formally, the algorithm is factorial in the number of runways, exponential in the number of classes, but polynomial in the number of aircraft in each class. It also provides an optimal makespan for the constrained dynamic interval over which the algorithm is iteratively run. Further work on mitigating the computational complexity of the dynamic version of the algorithm for multiple runways will involve creating lexically compact representations of the over-all state space, and the initial starting conditions of the algorithm.

References

[1] H.N. Psaraftis, "A Dynamic Programming Approach for Sequencing Groups of Identical Jobs," *Operations Research*, vol. 28, 1980.

[2] D. Harikiopoulo and N. A. Neogi, "Polynomial Time Feasibility Condition for Multi-Class Aircraft Sequencing on a Single Runway Airport," *IEEE Transactions on Intelligent Systems*, in press 2010.

[3] D. Harikiopoulo, "Class Dependent Sequencing Strategies for Landing Aircraft at Congested Airports," *M.S. Thesis*, University of Illinois, Urbana-Champaign, 2004.

[4] L.Bianco, P. Dell'Olmo, and S. Giordani, "Scheduling Models and Algorithms for TMA Traffic Management," in *Modelling and Simulation for Air Traffic Management*, L.Bianco, P. Dell'Olmo, and A.R.Odoni, Eds. 1997, pp. 139–167, Springer.

[5] R. De Neufville and A.R.Odoni, *Aiport Systems: Planning, Design, and Management*, Mc Graw-Hill, 2002.

[6] R.G. Dear, "The Dynamic Scheduling of Aircraft in the Near Terminal Area," FTL Report R76-9, Flight Transportation Laboratory, 1976.

[7] R.G. Dear and Y.S. Sherif, "An Algorithm for Computer Assisted Sequencing and Scheduling of Terminal Area Operations," *Transportation Research. Part A, Policy and Practise*, vol. 25, pp. 129–139, 1991.

[8] H.N. Psaraftis, "A Dynamic Programming Approach to the Aircraft Sequencing Problem," FTL Report R78-4, Flight Transportation Laboratory, 1978.

[9] C.S. Venkatakrishan, A. Barnett, and A.R.Odoni, "Landings at Logan Airport: Describing and Increasing Airport Capacity," *Transportation Sciences*, vol. 27, pp. 211–217, 1993.

[10] J.E Beasley, M. Krishnamoorty, Y.M. Sharaiha, and D. Abramson, "Scheduling Aircaft Landings - the Static Case," *Transportation Sciences*, vol. 34 (2), pp. 180–197, 2000.

[11] L. Bianco, S. Ricciardelli, G. Rinaldi, and A. Sassano, "Scheduling Tasks with Sequence Dependent Processing Times," *Naval Research Logistics*, vol. 35, pp. 177–184, 1988.

[12] L.Bianco, P. Dell'Olmo, and S. Giordani, "Minimizing Total Completion Time Subject to Release Dates and Sequence Dependent Processing Times," *Annals of Operations Research*, vol. 86, pp. 393–415, 1999.

[13] L.Bianco, P. Dell'Olmo, and S. Giordani, "Coordination of Traffic Flows in the TMA," in *New Concepts and Methods in Air Traffic Management*, L.Bianco, P. Dell'Olmo, and A.R.Odoni, Eds. 1999, pp. 95–124, Springer.

[14] J. Milan, "The Flow Management Problem in Air Traffic Control: A Model of Assigning Priorities at a Congested Airport," *Transp. Plan. Technol.*, vol. 20, pp. 131–162, 1997.

[15] C. Wu and R.E. Caves, "Research Review of Air Traffic Management," *Transport Reviews*, vol. 22, pp. 115–132, 2002.

[16] M. R. Garey and D. S. Johnson, *Computer and Intractability, a Guide to the Theory of NP-Completeness*, W.H.Freeman Company, 1979.